МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МОЭВМ

ОТЧЁТ

по лабораторной работе №2

по дисциплине «Алгоритмы и структуры данных»

Тема: Рекурсивная обработка иерархических списков

Студент гр. 8304		Кирьянов Д.И.
Преподаватель		Фирсов М.А.

Санкт-Петербург
2019

Задание.

Вариант №2

Подсчитать общий вес заданного бинарного коромысла bk, т. е. суммарный вес его гирек. Для этого ввести рекурсивную функцию.

Цель работы.

Решить полученную задачу, используя иерархические списки. Получение навыков работы с нелинейными структурами данных.

Описание алгоритма.

Сначала принимаемое значение записываем в строку. После чего при помощи стека проверяем на корректность подаваемых значений. Если ввод выполнен неправильно, то программа сообщает об этом и завершается. Если ввод выполнен верно, программа записывает введенное бинарное коромысло в список при помощи рекурсивной функции. Происходит запись либо узла, либо груза. Далее вызывается рекурсивная функция, проходящая по всему списку и считающая количество грузов. Если хранится узел, то рекурсия продолжается, иначе хранится груз. После выполнения данной функции программа завершается.

Описание функций программы:

1) `bool check(std::string array)`

Функция предназначена для проверки введенной строки.

array – подаваемая строка

2) `int read(std::string array, int index, binkor* kor)`

Функция предназначена для записи введенного бинарного коромысла в список.

array – подаваемая строка

index – индекс для возможности хождения по строке

kor – список, в который будет записано бинарное коромысло

3) `int count(binkor* kor)`

Функция предназначена для подсчета количества грузов в бинарном коромысле.

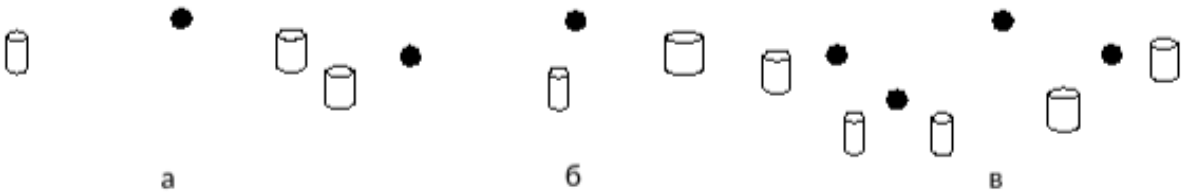
kor – список, в котором хранится бинарное коромысло

Описание структур данных:

1)
```
typedef struct binkor {
        int lenght;
        std::variant<std::pair<binkor*, binkor*>, int> system;
}binkor;
```

Структура предназначена для хранения бинарного коромысла.
length – длина плеча
system – переменная, хранящая в себе либо узел, либо массу груза

Графическая схема:



Выводы.

В ходе выполнения работы были изучены нелинейные структуры данных, был получен опыт работы с иерархическими списками. На мой взгляд данную работу можно было выполнить без использования иерархических списков.

Протокол

Тестирование:

Входные данные	Выходные данные
((5 10) (3 15))	2
((10 ((20 30) (30 40))) (50 60))	3

((10 ((20 30) (40((50 60) (70 80))))) (90 ((100 110) (120 130))))	5
((sdlf 10)(30 50))	13 2 3 44
((30 10)(10 5)))	Wrong Input
((3 7bb40 30))	Wrong Input
100	Wrong Input

Исходный код

lab2.cpp

```
#include <iostream>
#include <string>
#include <stack>
#include <variant>
#include <fstream>

typedef struct binkor {
	int lenght;
	std::variant<std::pair<binkor*, binkor*>, int> system;
}binkor;

bool check(std::string array) {
	std::stack <char> Stack;
	for (unsigned int i = 0; i < array.length(); i++) {
		if (array[i] == '(')
			Stack.push(array[i]);
		else if (array[i] == ')') {
			if (Stack.empty()) {
				return false;
			}
			Stack.pop();
		}
		else {
			if ((array[i] != ' ' && !isdigit(array[i]))||Stack.empty()) {
				return false;
			}
		}
	}
	if (!Stack.empty()) {
		return false;
	}
	return true;
}
int read(std::string array, int index, binkor* kor) {
	std::pair<binkor*, binkor*> side;
	if (isdigit(array[index])) {
		kor->lenght = std::stoi(array.substr(index));
	}
	while (isdigit(array[index])){
		index++;
	}
	if (array[index] == ' ')
		index++;
	if (array[index] == '(') {
		binkor* left = new binkor;
		side.first = left;
```

```
		kor->system = side;
		while (array[index] == '(')
			index++;
		index=read(array, index, std::get<std::pair<binkor*,binkor*>>(kor->sys-
tem).first);
	}
	else {
		kor->system = std::stoi(array.substr(index));
		while (isdigit(array[index])) {
			index++;
		}
	}
	if (array[index] == ' ')
		index++;
	if (array[index] == '(') {
		binkor* right = new binkor;
		side.second = right;
		kor->system = side;
		index++;
		index = read(array, index, std::get<std::pair<binkor*, binkor*>>(kor->sys-
tem).second);
		if (index == array.length() - 1)
			return 0;
		else
			index++;
	}
	if (array[index] == ')') {
		index++;
		return index;
	}
}
int count(binkor* kor) {
	int result = 0;
	if (std::holds_alternative<std::pair<binkor*, binkor*>>(kor->system)) {
		result += count(std::get<std::pair<binkor*, binkor*>>(kor->system).first);
		result += count(std::get<std::pair<binkor*, binkor*>>(kor->system).second);
		delete std::get<std::pair<binkor*, binkor*>>(kor->system).first;
		delete std::get<std::pair<binkor*, binkor*>>(kor->system).second;
	}
	else {
		result++;
	}
	return result;
}


int main(int argc, char* argv[]){
	std::string array;
	if (argc == 1) {
		std::getline(std::cin, array);
		if (!check(array)) {
			std::cout << "Wrong input" << std::endl;
			return 0;
		}
		binkor* kor;
		kor = new binkor;
		kor->lenght = 0;
		read(array, 0, kor);
		std::cout << count(kor) << std::endl;
		delete kor;
	}
	else {
		std::ifstream in(argv[1]);
		if (!in.is_open()) {
			std::cout << "Can't open file" << std::endl;
```

```
                return 0;
        }
        while (std::getline(in, array)) {
                std::cout << array << "\n";
                if (!check(array)){
                        std::cout << "Wrong input"<<std::endl;
                }
                else {
                        binkor* kor;
                        kor = new binkor;
                        kor->lenght = 0;
                        read(array, 0, kor);
                        std::cout << count(kor) << std::endl;
                        delete kor;
                }
        }
        in.close();
    }
    return 0;

}
```